SPIRE Photometer Interactive Analysis Package (SPIA)

Bernhard Schulz (IPAC/Caltech)

Contents

1. Intended Audience....1
2. Introduction and Scope....1
3. Overall Architecture and Tasks....2
4. Software Retrieval and Installation....3
5. Example Data Reduction Session....3
   5.1 Loading an observation....4
   5.2 Loading the calibration context....5
   5.3 Reprocessing the observation to Level 1....5
   5.4 Level 1 data inspection....8
   5.5 Reprocessing to Level 2....9
   5.6 Level 2 data inspection....10
   5.7 Saving the results....11
6. HSA Data Download....13
7. Software Download and Installation....13

1. Intended Audience

This document describes a piece of software that facilitates interaction with the SPIRE pipeline tasks under HIPE. It is intended for the regular astronomer that wants to improve the quality of the results over that of the standard pipeline products that the Herschel Science Archive provides by default. The reader is assumed to be familiar with the basic design of the SPIRE instrument and the Herschel satellite, the general flux calibration scheme and the pipeline description as given in the SPIRE Observers' Manual. It is further assumed that the reader has access to an installation of HIPE V4 and has familiarized himself at least with the help documents in the introductory section, in particular the HIPE Owner's Guide.

2. Introduction and Scope

The Herschel Interactive Processing Environment HIPE is a highly versatile platform. From the astronomer's point of view it provides as main ingredients mechanisms for i) data storage and retrieval, ii) a graphical user interface in the form of HIPE, iii) a scripting language, iv) a numeric library that can handle vectors and multidimensional arrays, and v) a set of pipeline scripts that reduce instrument data in a certain defined standard way. Although there have been several efforts that implemented convenience tools like image viewers, plotters for table datasets and products based on GUIs, most of the interaction with data remains script based. As scripts clearly allow the best versatility, they have also drawbacks like being error prone and somewhat difficult to remember if used only infrequently. GUIs typically allow less versatility but are easier to handle and provide better cues to remember functionality after not using the tool for longer periods of time.

So far there were only the official pipeline scripts that can be run as tasks that offer only a mimimum access to parameters of the data reduction, and typically astronomers and calibration scientists created their own derivatives to meet their specific needs. The SPIRE Photometer Interactive Analysis is an effort to provide a path in-between, combining the ease of use of a GUI interface with the versatility provided

by a modular design. The task framework of the HCSS, its large reservoir of functionality, and the automatic GUI support for tasks within HIPE made this project relatively easy to implement.

3. Overall Architecture and Tasks

The SPIA tasks are supposed to work interactively within a HIPE session environment. Central to the approach is the observation context, that is loaded into the session. This is the same observation context that is being loaded during the original pipeline scripts. The relevant tasks for a typical data reduction session, are shown as light blue boxes in Figure 1. There are tasks to bring data and calibration data from the Herschel Science Archive across the Internet into the local product store[1]. Other tasks load an observation context into the HIPE session or save a processed observation context or parts thereof back into the local store for safekeeping. Another IO task simply converts map products into standard FITS files that can be analyzed by standard astronomical applications like DS9 or Aladin.

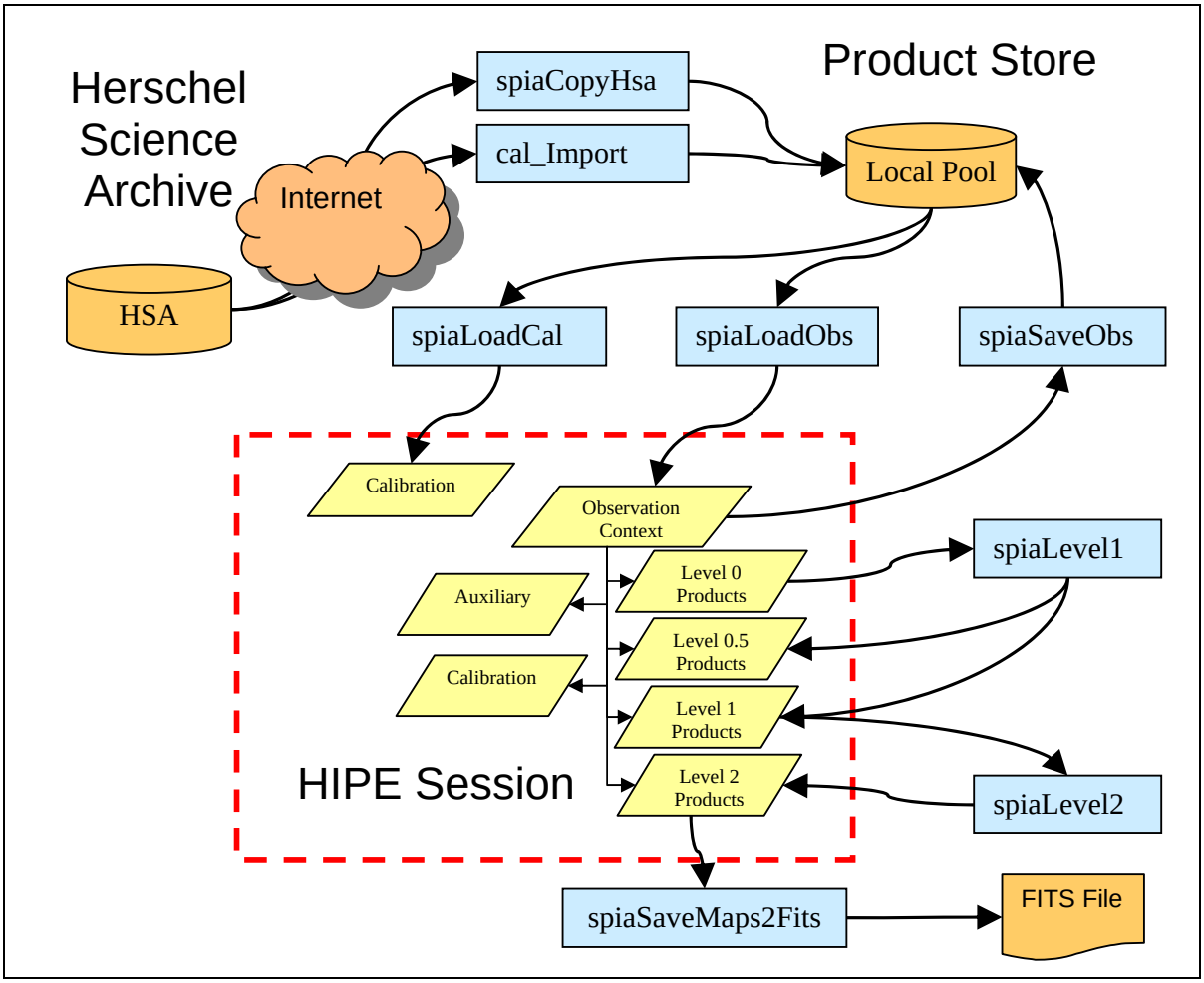

Fig 1: *An overview over the data flow when using the SPIA package. The data is extracted from the HAS via the internet into the local store, loaded into the session and saved back into the local store after processing with the tasks spiaLevel1 and spiaLevel2. The central data structure is the observation context with all its dependent products. Almost all tasks use the observation context as input, output or both.*

The two tasks that are effectively controlling the data processing from Level 0 via Level 0.5 to Level 1 and then from Level 1 to Level 2 are effectively user shells that call the identical pipeline tasks that are

[1] The cal_import task already existed but forms part of the SPIA data flow.

also called by the respective pipeline script. The main advantage of using these tasks is their graphical user interface that lays out all the parameters that are available and that can be changed. It should be clear though that doing so is entirely at the risk of the user. A thorough study of the relevant entry in the HIPE User's Manual or the SPIRE User's reference manual is essential to understanding the effect of any changes that are made to the default processing.

4. Software Retrieval and Installation

The software is currently distributed via the SPIRE-NHSC home page at:
https://nhscsci.ipac.caltech.edu/sc/index.php/Spire/DPsoftware
At the time of writing the package is at Version 0.6 and must still be considered an early beta version. The "Download" section contains links to this and earlier versions for historical reasons. It is always recommended to download the latest one. The software comes as zip-file and should be unzipped in its own directory. It contains a .py file that must be executed with the "Run all" button of HIPE before use so that the relevant tasks become known to the system.

5. Example Data Reduction Session

In this section a worked example is presented that should provide a simple way to become acquainted with the functionality of this package. The package consists effectively of just one Jython file that needs to be loaded first and executed with the "Run all" button (green double arrow) in HIPE. The procedure translates all the necessary classes and registers the tasks with HIPE so they show up in the Task-view of HIPE. It is recommended to set up the HIPE perspective in a fashion similar to the one shown in Figure 2 with the Editor-view, the Task-view and Outline-view on the right above each other, and the Variables-view to their left.

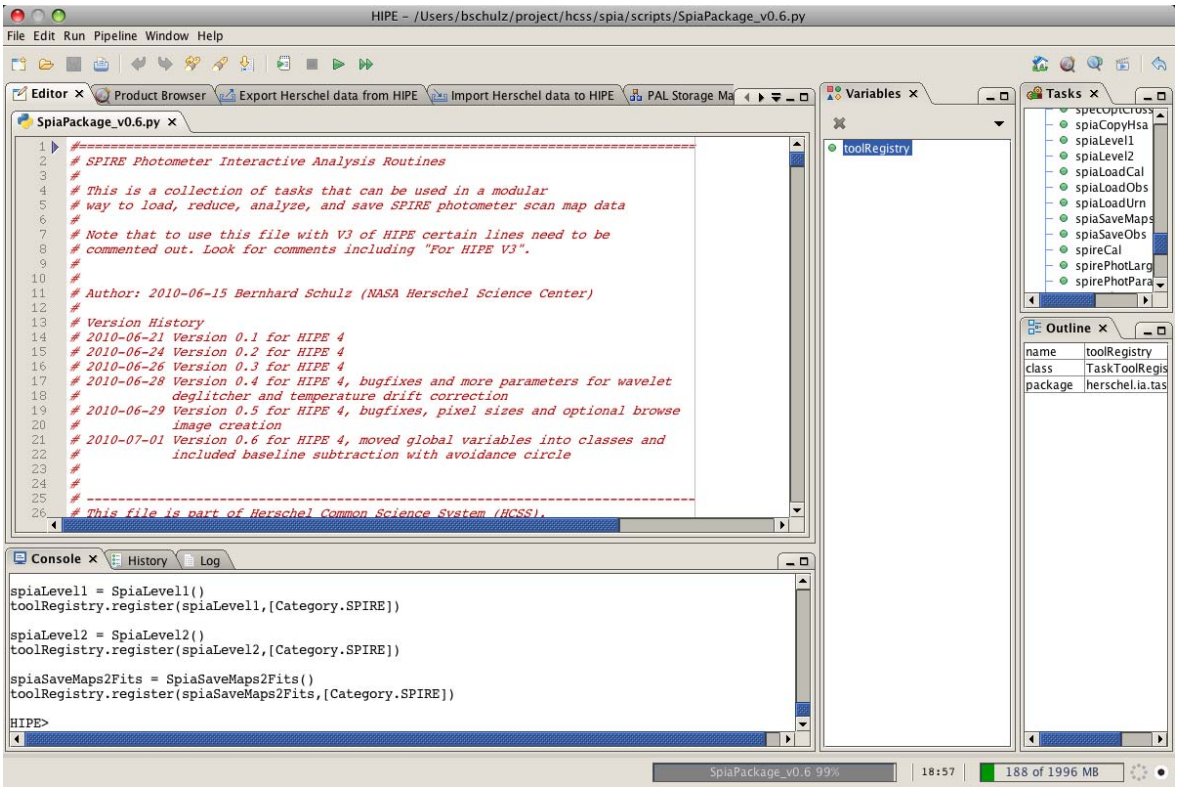
HIPE - /Users/bschulz/project/hcss/spia/scripts/SpiaPackage_v0.6.py
File Edit Run Pipeline Window Help
Editor
Product Browser
Export Herschel data from HIPE
Import Herschel data to HIPE
PAL Storage Ma
SpiaPackage_v0.6.py
SPIRE Photometer Interactive Analysis Routines
This is a collection of tasks that can be used in a modular
way to load, reduce, analyze, and save SPIRE photometer scan map data
Note that to use this file with V3 of HIPE certain lines need to be
commented out. Look for comments including "For HIPE V3".
Author: 2010-06-15 Bernhard Schulz (NASA Herschel Science Center)
Version History
2010-06-21 Version 0.1 for HIPE 4
2010-06-24 Version 0.2 for HIPE 4
2010-06-26 Version 0.3 for HIPE 4
2010-06-28 Version 0.4 for HIPE 4, bugfixes and more parameters for wavelet
deglitcher and temperature drift correction
2010-06-29 Version 0.5 for HIPE 4, bugfixes, pixel sizes and optional browse
image creation
2010-07-01 Version 0.6 for HIPE 4, moved global variables into classes and
included baseline subtraction with avoidance circle
This file is part of Herschel Common Science System (HCSS).
Console
History
Log
spiaLevel1 = SpiaLevel1()
toolRegistry.register(spiaLevel1,[Category.SPIRE])
spiaLevel2 = SpiaLevel2()
toolRegistry.register(spiaLevel2,[Category.SPIRE])
spiaSaveMaps2Fits = SpiaSaveMaps2Fits()
toolRegistry.register(spiaSaveMaps2Fits,[Category.SPIRE])
HIPE>
Variables
toolRegistry
Tasks
spiaCopyHsa
spiaLevel1
spiaLevel2
spiaLoadCal
spiaLoadObs
spiaLoadUrn
spiaSaveMaps
spiaSaveObs
spireCal
spirePhotLarg
spirePhotPara
Outline
name toolRegistry
class TaskToolRegis
package herschel.ia.tas
SpiaPackage_v0.6 99%
18:57
188 of 1996 MB

Fig 2: *HIPE panel after startup and configuration of recommended views, with SPIA script loaded and translated using the "Run all" button (green double arrow) in the menu at the top. The Tasks menu was opened and displays the newly addedtasks that all start with the prefix spia in the upper right.*

After translating the file and if everything went fine, a variable named toolRegistry will appear. The newly added tasks can be found in the Tasks-view. In the Tasks-view open “By Category” -> “Spire” where an alphabetical list of all tasks should appear that are registered SPIRE specific. All SPIA tasks begin with the prefix “spia” and are all found grouped together.

5.1 Loading an observation

We assume that we have a working system that already includes a local store with pools containing observations. To load an observation into the session we double-click on the task “spiaLoadObs” in the SPIRE Task-view. The GUI will open in the Editor view as shown in Figure 3 (left). Each input parameter has a field. In this case all are text fields for string input. Hovering with the mouse pointer over the name of a parameter will show a tool-tip giving more information about this item. Opening the additional tabs “Output” and “Info” brings out the full panel as seen in Figure 3 (right).

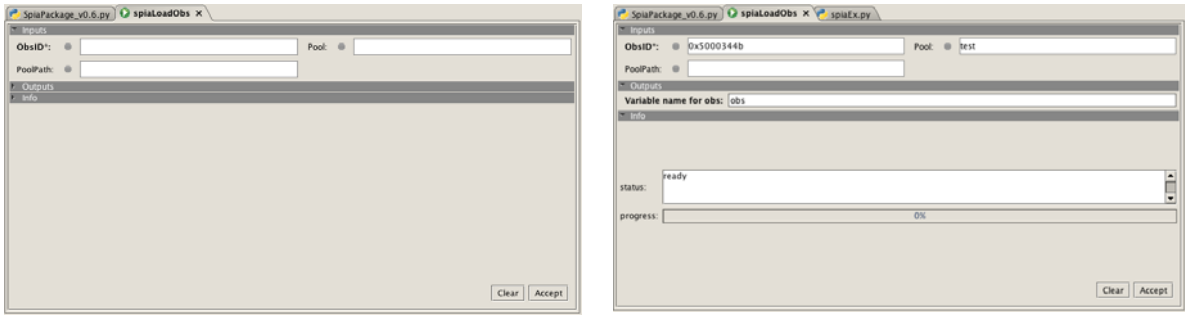

Fig 3: *The default GUI of the spiaLoadObs-task in initial configuration on the left, and with “Output” and “Info” panels opened and values for “ObsID” and “Pool” entered on the right.*

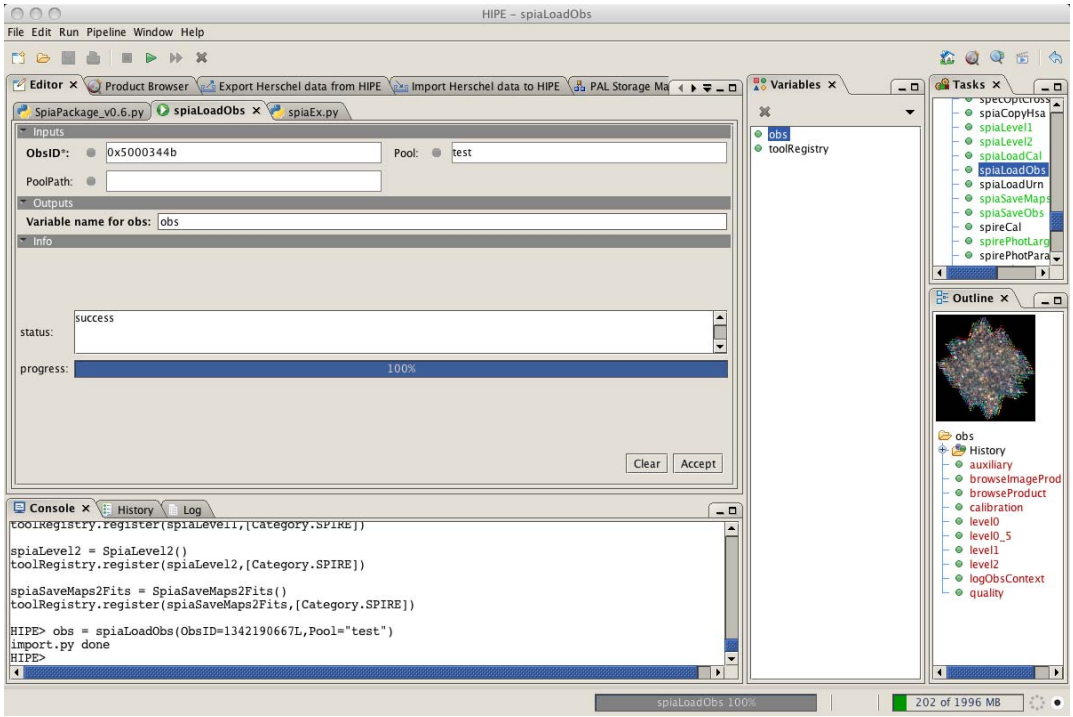

Fig 4: *The main HIPE panel after an observation context was loaded successfully. The equivalent command line is visible in the Console and the observation outline with browse image is displayed on the right while the variable for the observation context “obs” is highlighted in the Variables-view.*

Entering the observation identifier and the name of a pool provides the minimum amount of information the task needs to execute. In addition the task allows providing an optional path for the position of the local store on disk if it is not identical with the default path. Hitting the "Accept" button loads the observation context into the HIPE session. The new variable "obs" appears in the Variables-view, and the Outline-view shows the first level of products that are linked to the observation context as well as a browse image if available. The status panel in the GUI shows the message "success" and in the Console the equivalent script command line is printed. The same line will also appear in the Log-view and is very useful for recreating identical processing steps at a later time. All tasks in SPIA and HIPE in general are logged in this way (see Figure 4).

This is now a good time to inspect the observation. Typically one begins at the Level 2 inspecting the map products and processing logs, then making the way down to Level 1 to check the timelines and potential processing problems like missed glitches, issues with the temperature correction due to steps in the thermistor timeline etc.

5.2 Loading the calibration context

Since often the data products were retrieved a while ago, one may want to initiate a reprocessing of the observation right away using the newest calibration context and pipeline. To load the calibration context into the session, we activate the GUI of the task "spiaLoadCal". Generally there are two sources for the calibration context, a) the observation context itself, or b) the calibration context stored in the local store.

If we go for the newest calibration, we usually choose the calibration context in the local store, that can be downloaded from the server by executing the program cal_import. For that we would have to exit HIPE, run cal_import and re-start HIPE again. We assume that this has already been done. Then hitting the "Accept" button without any further parameter input will load the calibration context into the variable "cal".

To take the calibration context rather from the observation, the observation context needs to be provided to the task as input. This is not entirely intuitive at first. The observation context is an object and as such can not be entered as a simple string. HIPE provides a drag and drop method instead. Just pick the observation context from the Variables-view with the mouse pointer using the left mouse button and drag it over the round button that appears to the right of the variable name "obs" in the GUI, until a plus sign appears. Then drop the variable by releasing the left mouse button. If it went well, the button should now be green and to its right the variable name of the observation context should be shown. On the right of the GUI a drop-down menu allows to indicate whether to load from the observation or not. The default is "No" since in most cases one can expect the calibration context that came with the observation to be out of date. Selecting "Yes" requires the observation context to be provided as explained before and as shown in Figure 5.

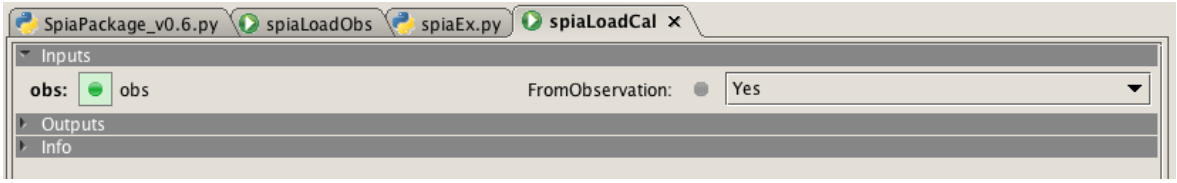

Fig 5: *The task "spiaLoadCal" in the configuration to load the calibration context from the observation. The green button to the left indicates that the observation context "obs" was provided. The drop-down menu on the right is switched to "Yes" indicating the choice to rather take the calibration context from the observation.*

5.3 Reprocessing the observation to Level 1

The menu that comes up when opening the task "spiaLevel1" is quite large and looks confusing at first. This is the largest number of parameter entries in a task within the SPIA and as such constitutes the heart

of the data reduction operation. The panel is usually larger than a screen and its upper and lower part are depicted separately in Figures 6a and 6b.

It should also be clearly stated here that the mere presence of a parameter doesn't indicate that it is essential to data reduction and needs to be changed. This task is meant to facilitate access to available parameters in order for them to be tested and examined and should be considered an expert level tool. Before experimenting with a parameter it is advisable to study the description of the respective pipeline module in the SPIRE Observer's Manual and the SPIRE User's Reference Manual. At this point it is probably fair to say that especially for the deglitchers, only a small part of the entire available parameter space has been tried. Eventually it is likely that good default values are found for all parameters, and only a few will need adjusting to special circumstances, but this is not the case yet.

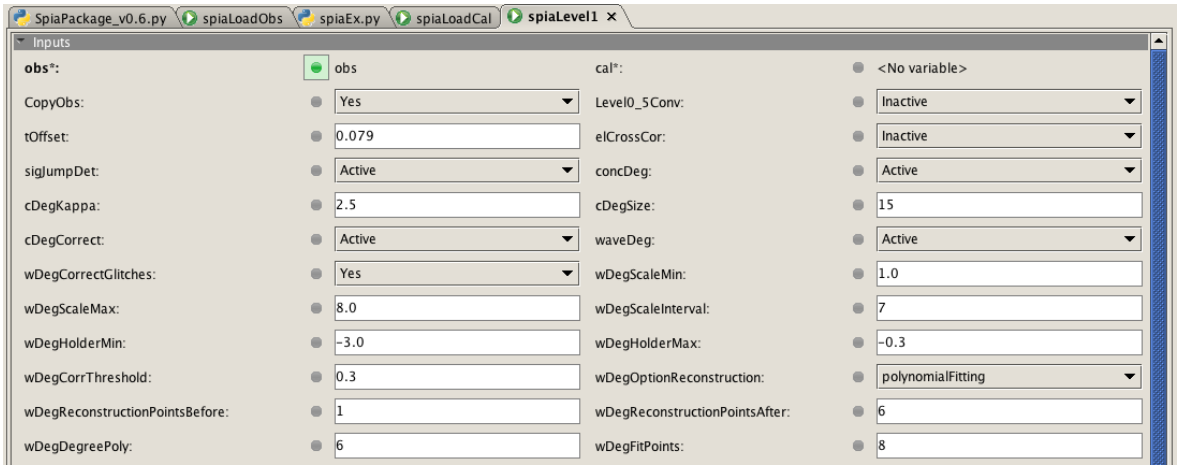

Fig 6a: *Upper part of the "spiaLevel1" processing task that repeats the data reduction starting from either Level 0 or Level 0.5 up to Level 1. This portion controls the creation of a separate observation context, the engineering conversion to Level 0.5, the correction of electrical cross correlation, the signal jump detection, the concurrent glitch deglitcher, and the wavelet deglitcher.*

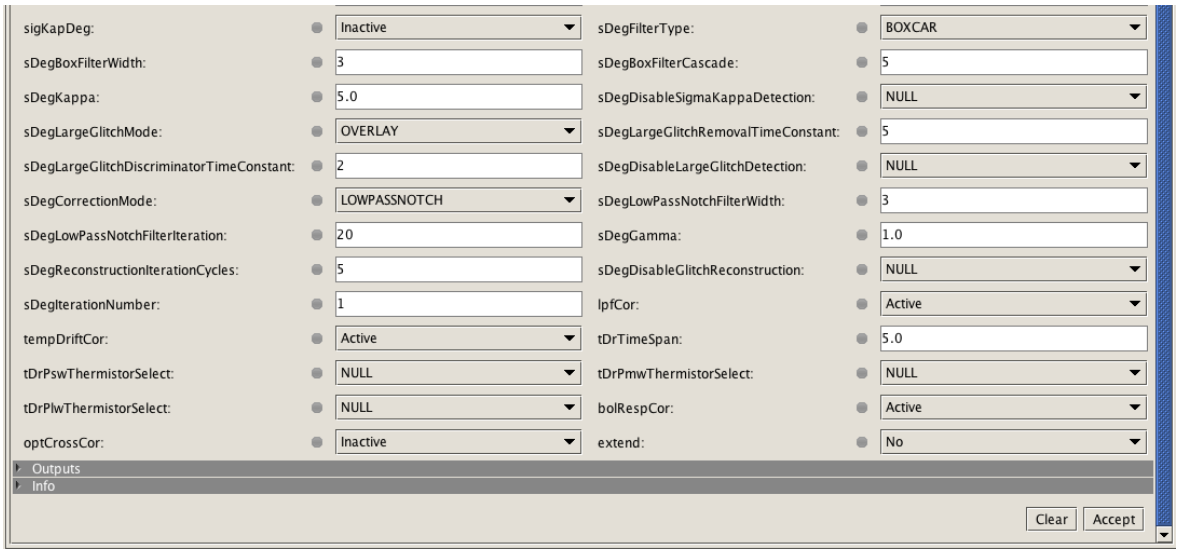

Fig 6b: *Upper part of the "spiaLevel1" processing task that repeats the data reduction starting from either Level 0 or Level 0.5 up to Level 1. This portion controls the Sigma Kappa deglitcher, thelowpass filter correction, the temperature drift correction, the bolometer response correction, the optical crosstalk correction and the inclusion of turnaround data.*

An exact description of every parameter in the Level 1 panel is beyond the scope of this document. Some additional information can be obtained from the tool-tips that exists for each parameter. However, a few parameters that control the data flow within the SPIA scheme will be explained here.

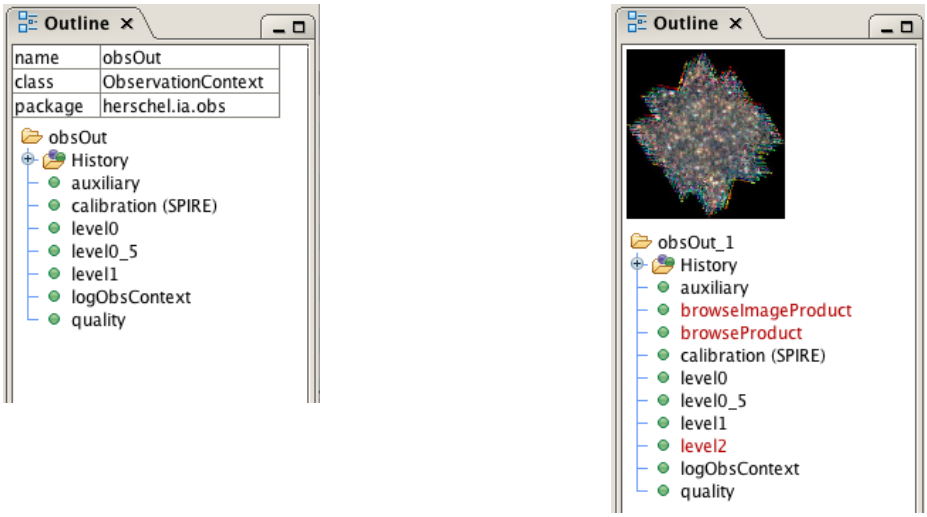

Fig 7: *Observation contexts after opting for a copy of the observation contex t(left) or after selecting to modify the original observation context (right). Note that Level 2 and browse product/image are not copied (left).*

To perform a successful reprocessing of SPIRE mapping data at Level 1, the task must be provided with an observation context and a calibration context. This is done by dragging and dropping from the Variables-view onto the respective round buttons in the task GUI as described before for the "spiaLoadCal" task. Both are mandatory input parameters, as indicated by the small asterisk close to the parameter name. The next parameter is called "CopyObs" and can be set via a pull-down menu to "Yes" or "No". It determines whether a new copy of an observation context should be created before the reprocessing. The copy includes the metadata, the calibration context, the auxiliary context, and the Level 0 context, and if not replaced by a reprocessed one, the Level 0.5 context. After pushing the "Apply" button, a lengthy command line with all parameter settings appears in the console and after the processing has finished, the newly reprocessed Level 1 context is placed in the copy of the observation context as shown in Figure 7 (left) and named by default "obsOut". Any other name can be assigned before processing in the "Output" panel. The name can also be changed after the fact by clicking once on the name of the observation context in the Variables-view and changing the name when it appears within a black frame after a second. If the variable name already exists when starting the task, a modified name is automatically used, like "obsOut_1" or "obsOut_2". If the answer to whether create a copy is "No", then the original observation context is used and only the reprocessed data products are replaced, i.e. in this case either Level 0.5 and Level 1, or Level 1 only. Note that the old Level 2 and browse product stay the same (see Figure 7 right).

All other parameters in the Level 1 task have default values that are currently accepted as generally working well. During processing messages appear in the Console-view indicating the building block that is being processed and warnings if modules are not executed because of corresponding selections in the "spiaLevel1" GUI. For instance by default the sigma kappa deglitcher is not selected and a warning will appear if the task is run in its default configuration.

5.4 Level 1 data inspection

When finished, the results can be inspected by double-click on the “level1” entry in the Outline-view. All Level 1 building blocks appear in a Photometer Scan Product-view. A single click on an icon for a building block to the left, produces the display of signal timeline data as shown in Figure 8.

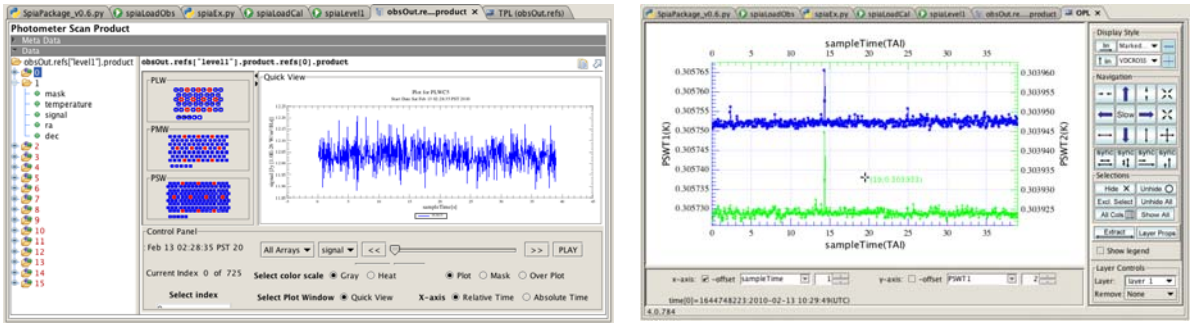

Fig 8: *Display of the first building block of Level1 data in the detector timeline viewer (left). The signals of all detectors can be inspected here for glitches and other artefacts. Via right-click on a building block icon other viewers like the Mask Editor and the Product Viewer are accessible for this data. An example of the Over Plotter being used to show temperature timelines of both PSW thermistors is shown on the right. This one, and other viewers are accessible by right clicking on one of the constituents of a building block icon in the view on the left.*

Right clicking on a building block icon allows two other viewers to be selected, the Mask Editor and the general Product Viewer. Clicking on the plus sign left of a building block icon, shows its constituents as seen in Figure 8 on the left below the selected building block. Right click on one of those products like signal or temperature makes several more viewers available like the Dataset Viewer, the Power Spectrum Generator, the Table Plotter and the Over Plotter. Figure 8 right shows an example of using the Over Plotter to show the temperature timelines of both PSW thermistors in the same diagram, revealing several concurrent and non-concurrent glitches that were apparently not found and restored. The Table Plotter is a simplified version of the Over Plotter that is easier to handle. The Dataset Viewer shows the actual numeric values a table dataset in terms of a spreadsheet (see Figure 9 left), and the Power Spectrum Generator (Figure 9 right) calculates a new dataset with a power spectrum of the timeline data, that will appear in the Variables-view and, being a Table Dataset, can be viewed again with tools like the Table Plotter, the Dataset Viewer etc.

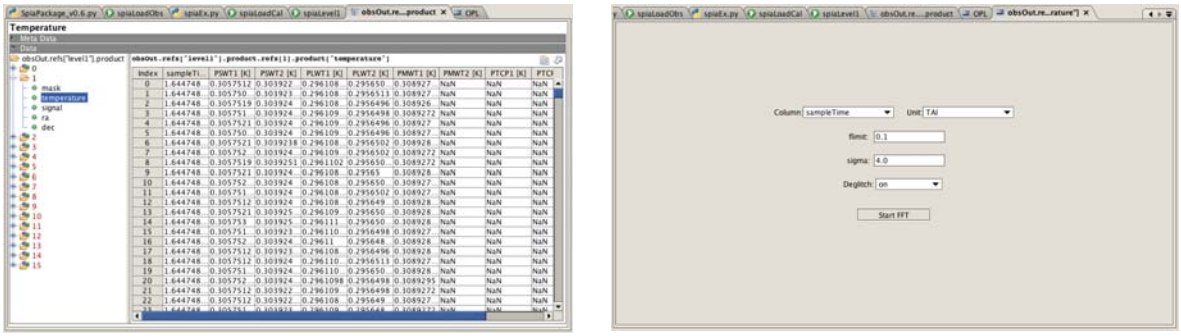

Fig 9: *TBW*

All these options for data inspection are available, as well as the use of Jython scripts since all datasets that appear in the Variables-view are available to the HIPE session.

5.5 Reprocessing to Level 2

The GUI of this task is launched in the same way as the others by double-clicking on the task "spiaLevel2" in the Task-view (see Figure 10). The menu is far less crowded but that is in part also a result of not having included yet all parameters that the called tasks actually offer. The two mandatory inputs are like in the "spiaLevel1" task, the observation context and the calibration context that are provided as usual through the drag and drop procedure described earlier. In this case the observation context for input is named "obsOut", which is the default name of the output product of the "spiaLevel1" task. The same selector "CopyObs" as in the previous task decides whether a new copied observation context should be used for the output or whether the task should just modify the input context. Although the default is set to "Yes", it is often useful to add the reprocessed Level 2 to an already existing observation context, which itself is a copy of the original that was produced by the "spiaLevel1" task. Independent of the actual choice, in the configuration shown, the default output of the task has the same default name and will be automatically changed to "obsOut_1".

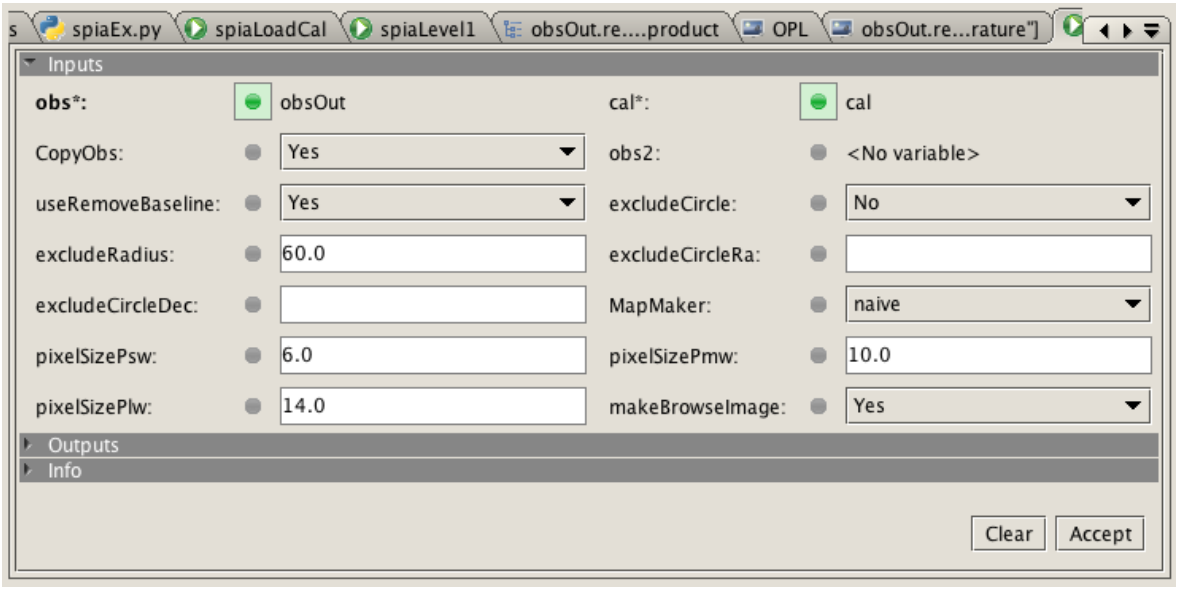

Fig 10: The "*spiaLevel2" GUI with observation context and calibration context already provided via drag and drop from the Variables-view. Not all parameters that exist at this level have been made available yet.*

There is another optional input "obs2" for another observation context. This option is intended specifically for parallel mode maps, where the orthogonal scan legs are in a different observation context. In such a case the two orthogonally scanned observations are processed separately with the "spiaLevel1" task and then both observation contexts are provided as input parameters to the Level 2 processing.

The Level 2 GUI gives the choice whether to use baseline removal, which is usually selected, unless the Level 1 context was pre-processed for that issue in a different way, perhaps by a custom script. If baseline removal is selected, additional four parameters provide an option to mask out a circular area around a position within the map that will not be used for determining the medians per scan. This can be useful the map is dominated by one bright source that distorts the distribution of fluxes.

As mapmakers the choice is offered between "Naïve" and "MADmap", and the pixel sizes in the map can be chosen differently from the defaults 6'', 10'', and 14'' for PSW, PMW and PLW respectively, if needed. Finally, there is a choice whether or not to generate the colour browse image, which currently is quite time consuming.

After hitting the "Accept" button to run the task, three map viewer windows appear, one for each wavelength. These are just to show immediately the result of the processing and can be closed at any time, without any impact on the results. The actual results are being linked into an observation context according to the selection made about producing a copy first or using the original. While the first map displays are already available, the generation of a colour map for the browse image is still ongoing, provided this option was selected. Processing is only finished after the circling dot in the lower right of the HIPE panel comes to a stop. It should also be noted that if the reader followed up to this point in his/her own HIPE session, and opted to not create a copy of the observation context for Level 2 processing because already the context named "obsOut" is a copy of the original, there will still appear a variable named "obsOut_1" in the Variables-view. This however is only a reference to the input observation context that can be deleted without deleting the result. It is good practise to do so through the right-click menu, to keep the number of variables down.

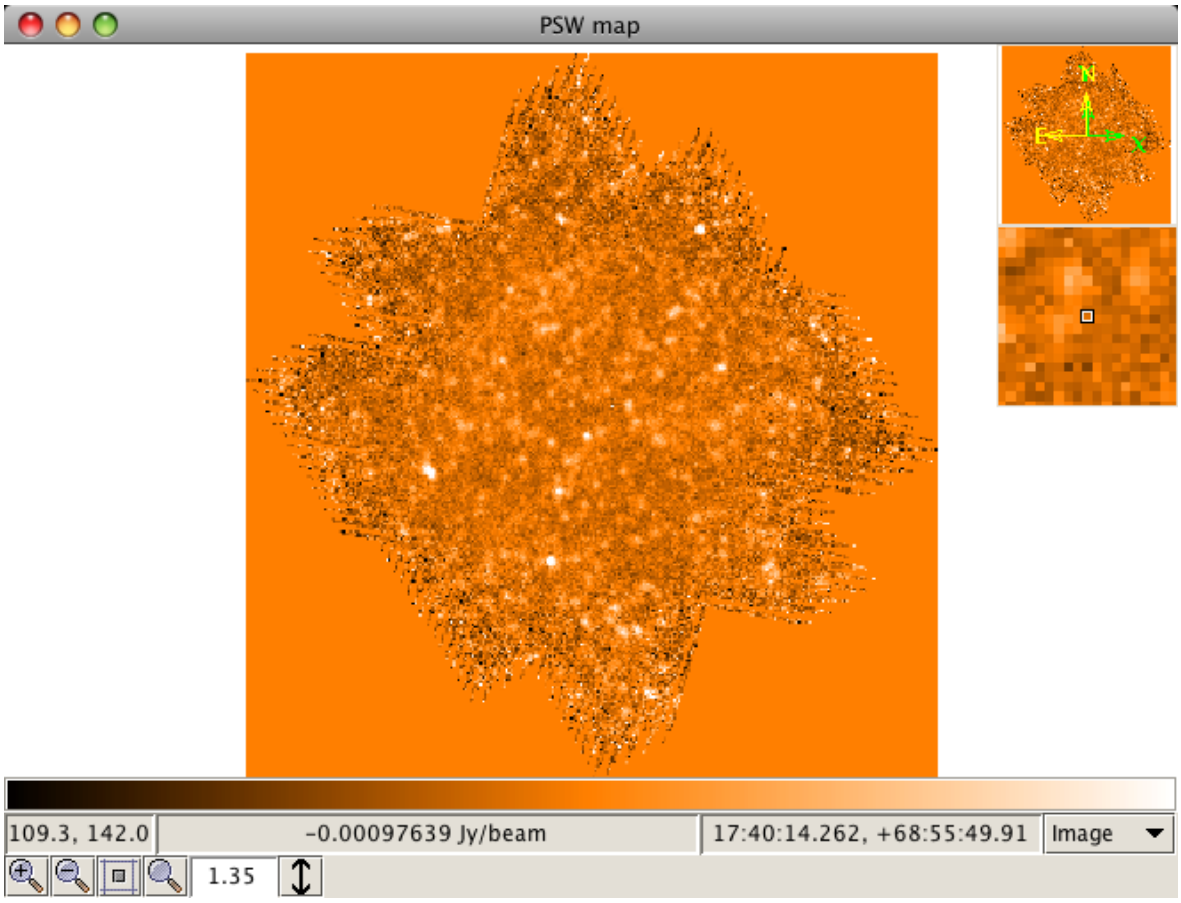

Fig 11 The *"spiaLevel2" task displays the maps generated for the three SPIRE detector arrays after completing in a map viewer. These windows are just informational and can be closed at any time.*

5.6 Level 2 data inspection

The popup windows that appear during Level 2 processing should already give a good idea about the result, as the Map Viewer itself has a wide range of functionalities going beyond the scope of this manual. However it should be mentioned that, the same tools and more are accessible from the level2 icon in the Outline-view. Double click or right click and selection of the Context Viewer, will bring up a panel like the one depicted in Figure 12. The left shows the components of the selected observation context. The context hierarchy can be opened down to the level of the array datasets that contains the image data, errors and coverage map. Right click on the array dataset allows choosing between Dataset Viewer (as used in Figure 12) and Image Viewer for Array datasets.

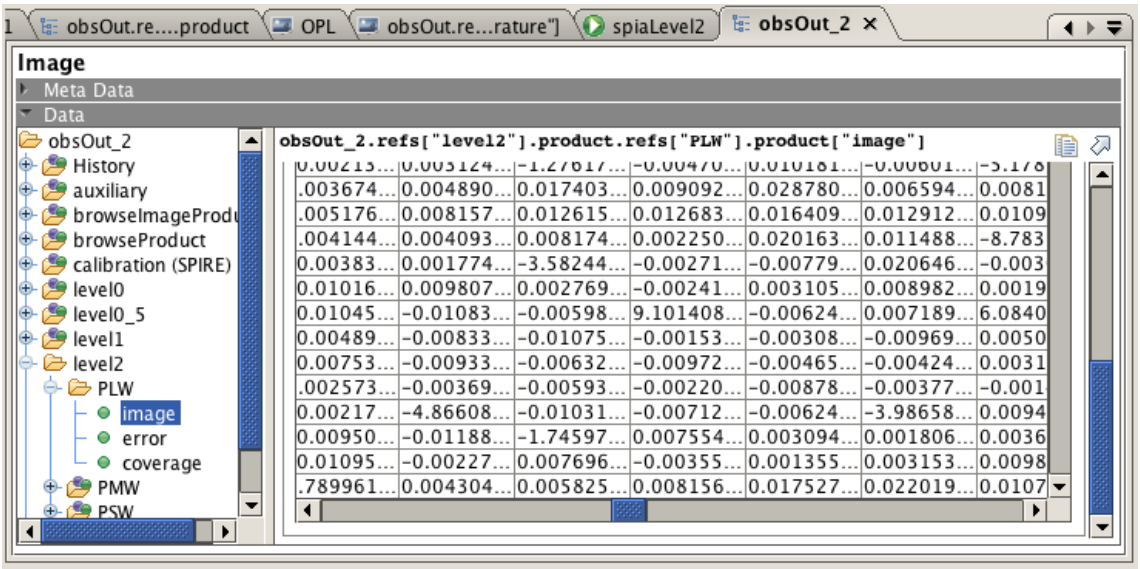

Fig 12 The *"spiaLevel2" task displays the maps generated for the three SPIRE detector arrays after completing in a map viewer. These windows are just informational and can be closed at any time.*

This is another point in the interactive data analysis cycle, where the user may well go back to the previous level and make an adjustment to his settings, or try a completely different choice of parameters. The system supports generation of multiple results of the same type, that can be held in memory of the session in parallel so direct comparison becomes easy. To make this effective, good management of the namespace is required, which is also well supported by the HIPE environment. Variables (mostly observation contexts), can be renamed directly within the Variable-view as described earlier.

5.7 Saving the results

The new observation contexts that are being produced during such an interactive analysis session are still residing in memory, at least in part, while the remainder is tucked away in a temporary storage pool on disk that is destroyed as soon as the HIPE session terminates. To keep at least the important results, a task names "spiaSaveObs" is provided that offers the default GUI shown in Figure 13.

The GUI is structured similar to that of the "spiaLoadObs" task and contains a field to enter the name of the target pool and optionally a path for that pool in case the default local store area should not being used. The output selection offers four choices: 1) Saving of the entire context, which saves a full copy of all associated products from Level 0 to Level 2, 2) saving of an observation context that contains only Level 1 products, 3) saving of an observation context with Level 2 products only, and 4) saving of an observation context that contains Level 1 and Level 2 products, but nothing else. The last three options are provided to save disk space in cases where the Level 1 processing is complete, and further work needs only to begin at Level 1 or 2.

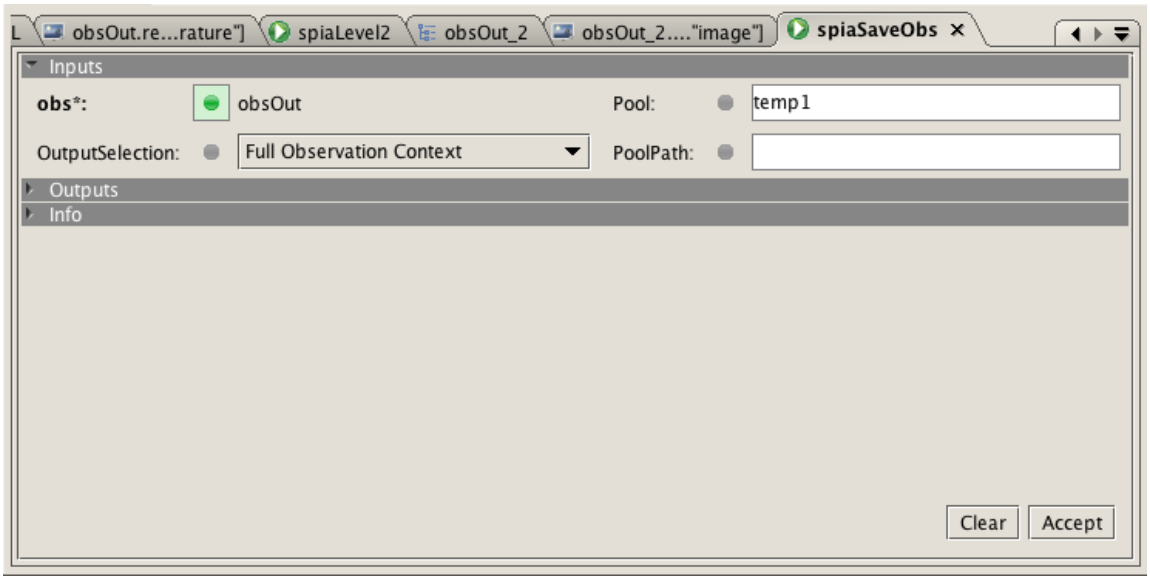

Fig 13 The default GUI of the "*spiaSaveObs" task. The observation context to save is mandatory input. Besides input fields for pool name and the optional path to the local store, the GUI provides a pull-down menu to choose the extent of the product levels to be saved.*

If the only product to be kept is Level 2, there is an alternative way of storage, that provides usually a preferable interface if subsequent analysis is to be done in other astronomical software packages. The task is named "spiaSaveMaps2Fits" and its default GUI is shown in Figure 14.

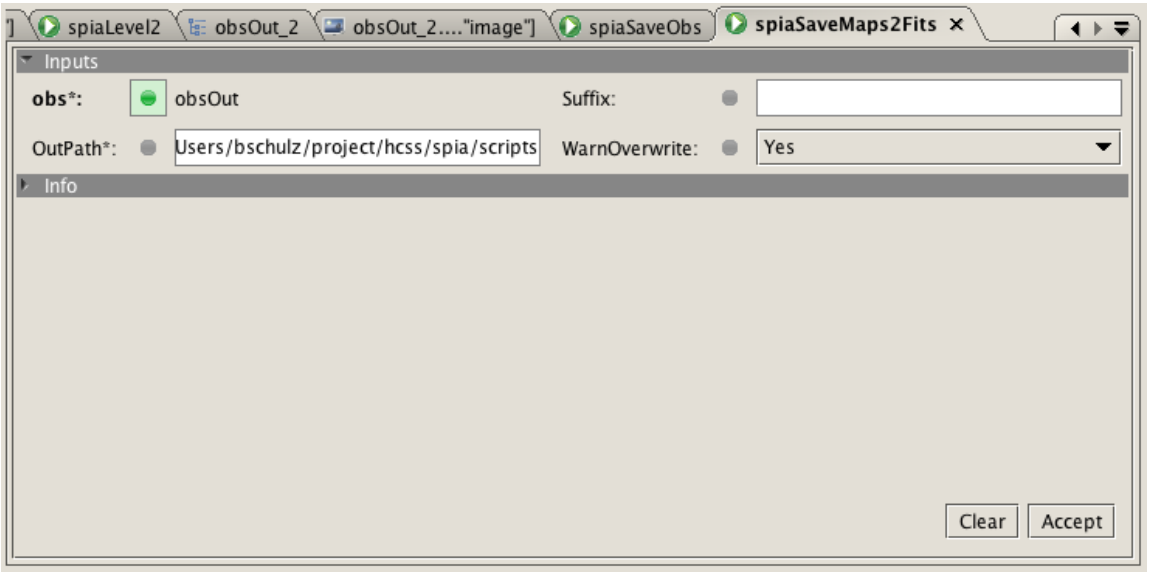

Fig 14 The default GUI of the "*spiaSaveMaps2Fits" task.*

This task accepts as input an observation context containing a Level 2 context. It saves three FITS files, one for each detector array, each containing three extensions representing flux map, error map, and coverage map. The filenames are generated from a user supplied suffix, the detector array name, and the observation identifier. The output path is set by default to be the location from where HIPE was started, and can be changed as needed. A switch that enables file overwrite warning completes this GUI.

6. HSA Data Download

In the example we started with an observation that already resided in the local store. The way of the product into the local store is usually via the search GUI of the Herschel Science Archive (HAS), a subsequent ftp transfer, unpacking and importing of the resulting data structure via the "Import Data to HIPE"-view. If the observation identifier is already known, the SPIA package provides an alternative method that uses direct access to the HSA via the Pool Access Layer (PAL). It queries the HSA for the observation, downloads it into the session and saves it immediately into the local store on disk. The GUI for this task is shown in Figure 15.

The input parameters are again the same as for the "spiaSaveObs" task, comprising of observation identifier, pool name, and optionally a path for the location of the pool. The variable name for the observation context as it will appear in the session is set to "obs" by default, however it can be changed as well. Analysis can in principle start directly from this context, however this requires maintaining the network link to the HSA for the entire time of the HIPE session until the results are saved, which may put a certain operational load on the HSA itself. It is generally better not to use this immediate observation context, but rather use the "spiaLoadObs" task to again get the observation context from the local store that has all links pointing onto the local disk rather than to locations across the internet.

Since the HSA maintains access control to its data, two properties need to be set that contain username and password. These are added to the the users.props file in the .hcss directory. The lines look like the following with <username> and <password> replaced by the real strings.

```
hcss.ia.pal.pool.hsa.haio.login_usr= <username>
hcss.ia.pal.pool.hsa.haio.login_pwd= <password>
```

This change can be performed in a normal text editor, but must be made while HIPE is not running.

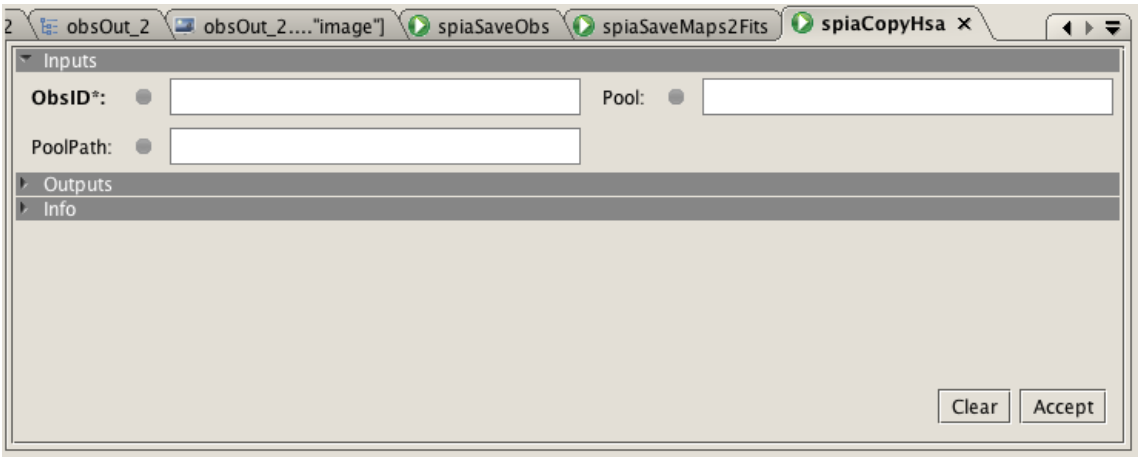

Fig 15 The default GUI of the "*spiaCopyHsa" task.*

7. Software Download and Installation

This software package is only a shell that pulls together the large set of functionalities that together make the Herschel Common Science System (HCSS) and the Herschel Interactive Processing Environment (HIPE) and fits into one Jython file. The file consists of several Jython classes that define HIPE tasks and need to be run and translated before they can be used. The end of the script contains a section that is actually executed, which registers the tasks with HIPE so they become visible in the Tasks-view.

The file, which comes as a packed .zip file can be retrieved via the internet from the observer support pages of the NASA Herschel Science Center at:

`https://nhscsci.ipac.caltech.edu/sc/index.php/Spire/DPsoftware`

It should be placed in a directory where HIPE is started or where most of the user's scripts are located. To make the SPIA tasks available in HIPE, the file must be opened and executed in HIPE by using the green "Run all" double arrow button. Instructions on how to install the file so that it is automatically executed every time HIPE starts will be provided at a later time.